TED UNIVERSITY

CMPE 492/SENG 492 Senior Project II

SyntaxSavior Final Report

3.05.2025

Team Members:

İrem Beşiroğlu Sofware Engineering

Yiğit Oğuzhan Kökten Computer Engineering

Yüksel Çağlar Baypınar Computer Engineering

1. Introduction

This final report details the design, implementation, and assessment phases of the SyntaxSavior project, an academic endeavor created to help rookie programmers in TED University's CMPE 113 course. The leap from theoretical programming principles to real coding might be especially difficult for first-year university students. SyntaxSavior bridges this gap by providing real-time, contextualized help via a lightweight development environment that integrates with Visual Studio Code (VS Code).

Novice programmers often struggle to translate classroom concepts into working code. In early courses such as CMPE113, students face repeated bugs and cryptic compiler messages, which hinder learning For example, research notes that many beginners are “plagued by recurring bugs, syntax errors and code smells” and find environment error messages “notoriously cryptic”. SyntaxSavior was motivated by this gap: it provides contextual, AI-driven feedback to help learners bridge theory and practice. By integrating automated analysis and feedback into the coding workflow, the system supports independent debugging and skill-building. Its scope includes assisting students during lab assignments (via a VS Code extension) and enabling instructors/assistants to manage lab instructions (via a web portal), thereby easing the transition from theory to practical coding.

The value of SyntaxSavior goes beyond the confines of a particular course. It demonstrates how intelligent instructional technologies may improve student interest and comprehension in computer science education, particularly in fundamental programming courses. This paper assesses the project's ultimate state from a systems engineering standpoint while also considering the larger global, sociological, economic, and educational consequences of incorporating such intelligent support systems in academic environments.

2. Final Architecture and Design

2.1. System Architecture

SyntaxSavior is a multi-tier system combining an IDE plugin, a web portal, and a backend. The **frontend** plugin is a Visual Studio Code extension written in TypeScript/JavaScript, using the official VS Code Extension API. The plugin monitors student Java code: it highlights syntax issues and sends code snippets to the backend when triggered by a submission command. The **assistant portal** is a web interface (built with standard web technologies) where teaching assistants upload or edit lab assignment instructions; it communicates with the backend via REST endpoints. The **backend** is a Spring Boot (Java) application that exposes a RESTful API. As a Spring Boot application, it auto-configures an embedded server to handle HTTP requests. The backend processes requests from both the plugin and portal: it authenticates users, fetches or stores lab instructions, and coordinates code analysis. When the plugin submits code, the backend forwards it to a separate **Python-based analysis engine** (running as a service). This engine uses a large language model (via the Gemini API) to interpret errors and generate feedback. In parallel, the backend queries a Milvus vector database to retrieve relevant course materials. Milvus is an open-source high-performance vector database designed for large-scale similarity search, which we use to semantically match the student's code or error against stored hints and documentation. All components follow REST architecture principles: resources are exposed via URIs and manipulated with standard HTTP methods. For example, the API treats code submissions and lab instructions as resources, in line with REST guidelines. Communication is via HTTPS calls (chosen for simplicity over more complex web sockets) with JSON payloads. Overall, SyntaxSavior's architecture is a clean separation of concerns: the VS Code front end (user interface), the assistant web front end (instruction management), and the backend services (business logic, AI analysis, and data storage).

2.2. Design Enhancements

Throughout development, several enhancements were made. The **user interface** was iteratively redesigned for clarity and usability; feedback from students led to improved layout and navigation

within the VS Code plugin. Code snippets sent to the backend now use secure transport (HTTPS/TLS) to protect student work. Indeed, the system's security controls enforce encrypted data transfers and user authentication, so code and user credentials are never sent in plaintext. The **code analysis engine** was refined to recognize more solution patterns: for example, the vector database was expanded with additional lab examples, and the matching algorithm was tuned to better identify logically correct variants of common problems. Finally, a new "**submission readiness**" feature was added. The backend now evaluates code complexity and error counts to give students a readiness indicator – warning them if the solution contains many errors or is likely incomplete. These improvements (secure communication, smarter pattern matching, and readiness detection) were driven by testing and feedback, ensuring the final design is robust and user-friendly.

3. Engineering Impact

3.1. Global and Societal Impact

SyntaxSavior contributes to educational equity by making programming help more widely accessible. By automating preliminary feedback, it can serve many students without requiring one-on-one tutoring for every error. In regions with good connectivity, such AI tools can "bridge gaps in subject expertise" and connect learners to quality instruction. In this way, SyntaxSavior aligns with global education goals: it supports quality education (UN SDG 4) by providing scaffolding for novices. On a societal level, improving coding proficiency addresses the workforce skill gap in technology fields. However, care must be taken to ensure pedagogy guides AI use; UNESCO warns that educational AI should be steered by pedagogical principles and human values. SyntaxSavior is designed with these principles in mind, aiming to enhance (not replace) instruction and to promote student learning rather than shortcuts.

3.2. Economic Impact

Economically, SyntaxSavior could reduce instructional costs by lightening the grading and tutoring burden on faculty and teaching assistants. Automating common feedback frees instructors to focus on deeper teaching activities. As an educational technology, it may spawn new edtech

services or roles (for example, maintaining and localizing the system). In the long run, by improving student coding skills early on, the project may boost workforce readiness and productivity in the software industry. On the other hand, initial development costs (engineering time, cloud resources for AI queries) must be weighed. Ultimately, the tool is intended to be open and cost-efficient: we built it with readily available technologies and focused on reuse of pre-trained models (via the Gemini API) rather than expensive training in-house.

3.3. Environmental Impact

The environmental footprint of SyntaxSavior is mostly in its computational components. Running the Spring Boot server and the Milvus database on campus servers has a modest energy cost. More significantly, querying a large language model (LLM) has an energy impact. For context, training a model like GPT-3 consumed on the order of 1,287 MWh and emitted ~552 metric tons of CO_2. By contrast, SyntaxSavior uses a pre-trained model via API, so the heavy training energy is externalized (Google handles that). Each inference call still consumes some compute on the cloud, which contributes to the carbon footprint. We mitigate this by batching requests and caching common results when possible. In the broader view, an in-classroom AI tutor likely consumes far less energy per student hour than having every student travel or every assignment graded manually. Nonetheless, we monitor usage and aim to run services on efficient infrastructure to minimize energy use. Additionally through digital content delivery and reduced need for printed materials, the system supports sustainable educational practices.

4. Contemporary Issues

4.1. Use of AI in Education

AI-powered learning tools are an emerging trend in education. Such tools can offer personalized, on-demand help and free educators from repetitive tasks. For example, UNESCO notes that AI is increasingly used to automate grading and administrative tasks, potentially easing teachers' workload. SyntaxSavior embodies this promise by delivering instant feedback on code. However, AI in education also raises questions: there is not yet conclusive evidence that generative AI improves learning outcomes. We must ensure that AI suggestions encourage understanding, not

just correct answers. Our design intentionally frames feedback in educational terms (e.g. explaining the *reason* for a syntax error) and avoids giving outright solutions, in line with best practices in AI-in-education research.

4.2. Cheating and Generative AI

The rise of powerful AI (e.g. ChatGPT) has made cheating a hot-button issue. Students could misuse generative AI to automatically write code answers. Experiments have shown that while AI tools can solve problems quickly, students learn less: in one MIT study, students using ChatGPT solved a task fastest but later "remembered nothing". SyntaxSavior's approach is different: rather than generating solutions, it explains and guides. Nevertheless, we built into the system and intend to work in tandem with already existing anti-cheating measures (e.g. input validation, plagiarism checks). For instance, code submissions are logged and any attempt to bypass the plugin (e.g. submitting random output) is flagged. By focusing on explanation rather than answer-generation, SyntaxSavior aims to deter misuse and promote learning integrity.

4.3. Accessibility and Digital Divide

SyntaxSavior addresses the digital divide by making coding tools available to all users regardless of their hardware limitations. The application runs smoothly on school lab computers with mid-to-low tier hardware, requiring significantly less system resources than alternatives like Eclipse IDE that are installed in our educational environment. This lightweight design ensures students and educators can access professional development tools without the need for equipment upgrades or specialized hardware. SyntaxSavior works across Windows, macOS and Linux platforms due to the cross platform of VS Code, allowing users to maintain consistent workflows regardless of which operating system they use at school or home. The tool's modest bandwidth requirements make it practical for users in lab environments where the load on network may be high due to the number of students actively using it. By integrating seamlessly with VS Code, SyntaxSavior creates a bridge between entry-level programming environments and industry-standard tools, helping users transition between platforms based on their current resources without disrupting their learning or development process.

5. New Tools and Technologies Used

- **VS Code Extension Development**: Developed in TypeScript using the VS Code Extension API, which provides hooks for creating commands, editors, and UI within the IDE.
- **Spring Boot:** A Java framework that simplifies building RESTful web services; its @SpringBootApplication annotation auto-configures an embedded server and components. We used Spring MVC and Spring Security for API endpoints and authentication
- **Milvus Vector Database:** We use Milvus (an open-source high-performance vector database) to store embeddings of lab instructions and code examples. Queries from the backend find the nearest vectors to a given code snippet, enabling semantic retrieval of relevant hints. Milvus's Python client API makes integration straightforward.
- **Python LLM Engine**: The Python LLM Engine is a dedicated backend service responsible for invoking the LLM API to assist with educational guidance and code-related feedback. It serves as the intelligence layer behind the SyntaxSavior assistant. We utilize RESTful API calls to access LLM's chat/completions endpoints with carefully designed system and user prompts tailored to CMPE113 coursework. To ensure that the responses are highly relevant and aligned with the course content, we fine-tune the behavior of Gemini using custom prompts constructed dynamically from context-retrieved documents stored in a Milvus vector database. This database contains lecture notes, lab instructions and tasks, Course textbook excerpts and additional curated programming explanations.
- **Socket/REST Communication**: The plugin and portal communicate with the backend using HTTPS (chosen over WebSockets for simplicity). Similarly, the backend and Python engine communicate over a REST API. This decision leveraged existing REST guidelines and avoids the overhead of maintaining WebSocket state.

6. Background Research and Resources

Our design was informed by research in programming education and by vendor documentation. We reviewed education research showing that contextual, elaborated feedback can aid novices,

and used taxonomies of student errors (e.g., Becker *et al.*, 2019) to categorize problems. Classic pedagogy (such as *Productive Failure* models) influenced our feedback style. On the technical side, we studied the [Java Language Specification] and Oracle's Java API docs to interpret common compiler messages. Vendor documentation guided implementation: for example, the IEEE 1016-2009 standard was used as a blueprint for structuring our design documents, and Spring Boot and VS Code API guides provided usage examples. We also consulted the Gemini API documentation for model usage and the Milvus documentation for embedding indexing. Altogether, these resources ensured our technical choices were sound and our pedagogical approach was research-based.

7. Test Results

We devised test cases covering all major components. The following table summarizes representative test cases (as documented in the Test Plan) and outcomes:

ID	**Feature/Component**	**Test Case**	**Expected Outcome**	**Result**	**Notes**
TC-PL-01	VS Code Plugin UI	Verify plugin menus, buttons, and prompts	UI elements render correctly; clickable actions work.	**Pass**	Minor UI tweaks needed on theme.
TC-PL-02	Plugin-Backend Communication	Student submits code via plugin	Code is sent to server; success message received.	**Pass**	
TC-BE-01	Backend API (REST)	GET/POST on code submission and instruction endpoints	Correct HTTP status and JSON response.	**Pass**	

ID	Feature/Component	Test Case	Expected Outcome	Result	Notes
TC-AI-01	Python Analysis Engine	Submit code with syntax error	AI returns an appropriate feedback message.	**Pass**	Accuracy of message checked by graders.
TC-DB-01	Vector DB Retrieval	Query Milvus with a known code snippet	System returns related lab hints/materials.	**Pass**	
TC-UR-01	User Roles	Attempt student action with admin credentials	Access denied (student-only function).	**Pass**	Role-based checks working.
TC-UR-02	User Roles	Instructor uploads lab instructions via portal	Instructions stored in DB; accessible to students.	**Pass**	
TC-SEC-01	Security	Submit malicious input or attempt SQL injection	Input is sanitized; request is rejected or safe.	**Pass**	No vulnerabilities found.
TC-SEC-02	Encryption	Inspect network traffic during submission	Payload is encrypted.	**Pass**	Verified with SSL monitoring.
TC-ERR-01	Error Handling	Force network/server failure during submission	Plugin shows error notification; no crash.	**Pass**	

In summary, all critical test cases passed. The system correctly handled code submission, feedback generation, data retrieval, and security checks. User role enforcement and error handling worked as intended. Minor UI issues (theme alignment) were noted and fixed. Overall, the tests validated that SyntaxSavior meets its functional and reliability goals.

8. Ethical Considerations

The ethical responsibilities are considered by the team while developing this project. The application of artificial intelligence (AI) in the classroom is not only bringing technological benefits but also causes a series of ethical responsibilities that must be explicitly laid out to ensure fairness, integrity, and student's learning. SyntaxSavior is developed with a strong emphasis on educational ethics, particularly in the field of computer science education where help can so quickly become cheating.

- **Academic Integrity**
  One of the principal ethical concerns in the development of AI-based learning aids is maintaining academic integrity. SyntaxSavior is engineered to provide not direct code solutions, but step-by-step conceptual hints and curriculum-aligned explanations. In so doing, it promotes learning through guided discovery rather than spoon-feeding solutions. The system includes hint frequency limits within its design and identifies repetitive or passive behaviors indicative of over-reliance. All output is designed to promote understanding, and functions such as similarity detection and session logs are used to discourage plagiarism.

- **Data Privacy and User Consent**

  Administering user information—student-crafted source code, to be precise—requires strict compliance with data protection provisions. SyntaxSavior ensures complete data transmission encryption between the backend and plugin, utilizing industry-wide standards (HTTPS with TLS 1.3). Persistent storing of no student data takes place without consent being explicitly asked for, and personally identifiable information gets anonymized on checking for tests or analysis purposes. Later developments intend to include manifest consent practices as well as policies for retaining information in compliance with GDPR and parallel frameworks.

- **Transparency and Explainability**

  One challenge in educational AI is the 'black box' nature of many algorithms. SyntaxSavior mitigates this by using deterministic, explainable rule-based components in tandem with the AI model. Feedback includes rationale statements (e.g., "This error may occur because variable

X was not initialized"), helping students understand not only what went wrong, but why. This fosters metacognitive development and avoids blind trust in the system's suggestions.

- **Human-Centered Design and Teacher Augmentation**

  SyntaxSavior is not intended to replace instructors, but to augment their capacity to support students. Ethical deployment of AI in education should empower teachers by automating repetitive tasks (e.g., syntax checks), freeing them to focus on deeper mentorship. The assistant portal is specifically designed to keep humans in the loop: instructors can give lab guidance by uploading the specific lab's instruction pdf.

9. Future Enhancements

Possible improvements include **LMS Integration** (linking SyntaxSavior with university learning management systems to streamline authentication and possibly auto-fetch assignments), **IntelliJ Support** (developing an equivalent plugin for the IntelliJ IDE), **Enhanced Personalization** (adapting hints to individual student history), and **Gamification Features** (badges or progress tracking to motivate learners). Other ideas are expanding to additional programming languages and adding a framework that allows peer-to-peer collaboration. Finally, ongoing refinement of the AI feedback (e.g. multi-turn clarification dialogs) and scaling infrastructure (for more courses) may be planned as the project matures.

10. Conclusion

SyntaxSavior was conceived as a means to efficiently solve the obstacles experienced in CMPE 113 programming laboratories. It effectively demonstrates its value as an intelligent tutoring aid for introductory programming, efficiently addressing the aforementioned obstacles in an innovative way. By combining a VS Code plugin, an assistant portal, and a robust backend architecture (Spring Boot + Python AI + Milvus), the system delivers real-time, contextual guidance to novice programmers while reinforcing teaching objectives through structured feedback. A key strength of the platform lies in its ability to transform opaque compiler errors into explanatory feedback that aligns with course objectives, bridging the gap between theoretical understanding and practical application. As an academic project, SyntaxSavior not only hopes to enhance students' programming capabilities but also serve as a valuable learning platform for ourselves as the developers, following software engineering standards such as IEEE 1016 for design documentation and implementing modern architectural practices. In conclusion,

SyntaxSavior demonstrates clear pedagogical value by acting as an intelligent, context-aware learning helper that reinforces conceptual understanding, while simultaneously laying the groundwork for further research in AI-enhanced engineering education.

11. References

1. IEEE 1016-2009, *IEEE Standard for Information Technology – Systems Design – Software Design Descriptions* (https://en.wikipedia.org/wiki/Software_design_description#:~:text=IEEE%201016)
2. Zalando, *RESTful API and Event Guidelines (https://opensource.zalando.com/restful-api-guidelines/#:~:text=Comparing%20SOA%20web%20service%20interfacing,and%20can%20be%20manipulated%20via)*.
3. Visual Studio Code Extension API Reference (https://code.visualstudio.com/api/references/vscode-api#:~:text=VS%20Code%20API .
4. Milvus Documentation (Open-source Vector Database) (https://milvus.io/#:~:text=Milvus%20is%20an%20open,vectors%20with%20minimal%20performance%20loss).
5. Wang *et al.*, "Learning From Errors" (Frontiers in Education, 2021) – on novice error comprehension (https://pmc.ncbi.nlm.nih.gov/articles/PMC8669433/#:~:text=error%20messages%20contain%20detailed%20information,missing%20a).
6. Giri & Richard, *ICLS 2023* – "It's a Little Frustrating, but Fun" (on novice debugging challenges) (https://www.researchgate.net/publication/375408699_It's_a_Little_Frustrating_but_Fun_Supporting_Novice_Programmers'_Learning_Through_Unscaffolded_Problem-Based_Designs).
7. Shein, "The Impact of AI on Computer Science Education" (Commun. ACM, 2024) (https://cacm.acm.org/news/the-impact-of-ai-on-computer-science-

education/#:~:text=One%20group%20was%20allowed%20to,the%20task%20down%20into%20components).

8. UNESCO, “Use of AI in Education” (article, 2023) (https://www.unesco.org/en/articles/use-ai-education-deciding-future-we-want); UNESCO Global Partnership blog (2022) (https://www.globalpartnership.org/blog/bridging-gap-how-technology-can-mitigate-global-teaching-crisis#:~:text=Where%20robust%20internet%20access%20exists%2C,subjects) (on education tech and access).
9. Smith *et al.*, *“The Carbon Emissions of Writing and Illustrating”* (Nat. Sci. Rep. 2024) – figures on AI training energy (https://www.nature.com/articles/s41598-024-54271-x#:~:text=on%20the%20environmental%20impact%20of,5%20metric%20tons%20of).